ARMY RESEARCH LABORATORY

Implementation of Spread Spectrum Image Steganography

by Frederick S. Brundick and Lisa M. Marvel

ARL-TR-2433

March 2001

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5067

ARL-TR-2433 **March 2001**

Implementation of Spread Spectrum Image Steganography

Frederick S. Brundick and Lisa M. Marvel
Computational and Information Sciences Directorate, ARL

Abstract

Steganographic techniques are useful to convey hidden information by using various types of typically-transmitted multimedia data as cover for concealed communication. Spread Spectrum Image Steganography (SSIS) is a data-hiding/hidden-communication method that uses digital imagery as a cover signal. This report describes an SSIS prototype system for embedding messages in images and extracting messages from stegoimages. The components, which were written in the Java and C programming languages, were kept modular to provide a workbench for further experimentation. We discuss experiments that were performed with the system, along with possible avenues of future research to improve the SSIS process.

Table of Contents

INTENTIONALLY LEFT BLANK.

List of Figures

List of Tables

INTENTIONALLY LEFT BLANK.

1. Introduction

The prevalence of multimedia data in our electronic world exposes a new avenue for communication using digital steganography. Steganographic techniques are useful to convey hidden information by using various types of typically-transmitted multimedia data as cover for concealed communication. The inability to detect the hidden data, perceptually or by computer analysis, is paramount for surreptitious operation.

There are many applications for techniques that embed information within digital images. The dispatch of hidden messages is an obvious function, but today's technology stimulates even more subtle uses. In-band captioning, such as movie subtitles, is one such use where textual information can be embedded within the image. The ability to deposit image creation and revision information within the image provides a form of revision tracking as another possible application of digital steganography. This avoids the need for maintaining two separate media, one containing the image itself and one containing the revision data. Authentication and tamperproofing as security measures are yet other functions that could be provided. Digital image steganographic techniques can also provide forward and backward compatibility by embedding information in an image in an imperceptible manner. If a system has the ability to decode the embedded information, new, enhanced capabilities could be provided. If a system did not have the capability to decode the information, the image would be displayed without degradation, leaving the viewer unaware that the hidden data exist. These are but a few of the possible uses of image steganography.

Spread Spectrum Image Steganography (SSIS), is a data-hiding/hidden-communication method that uses digital imagery as a cover signal. SSIS provides the ability to hide and recover, error free, a significant quantity of information bits within digital images, avoiding detection by an observer. Furthermore, SSIS is a blind scheme because the original image is not needed to extract the hidden information. The proposed recipient need only possess a key in order to reveal the hidden message. The very existence of the hidden information is virtually undetectable.

This report describes an SSIS prototype system for embedding messages in images and extracting messages from stegoimages. Portability was a major consideration when designing this application. The research and initial experimentation had been performed in MATLAB and C under the UNIX operating system, while the target system for the prototype was a laptop running Windows 95. This was achieved by rewriting the MATLAB portions in ANSI C and by using Java for the graphical user interface (GUI).

The initial intention for building the prototype was to provide a vehicle by which to demonstrate the concepts of SSIS. However, the prototype was found to be very useful from a research standpoint. It was important to build flexibility into the system to provide a workbench for further experimentation. Each of the major components was written as a separate program, which also allowed multiple researchers to work on different areas of the problem. In addition, tests could be performed without having to modify the prototype.

After providing background information on the steganography process, we will discuss the components of the SSIS prototype and how they work together to function as an intact system. The methods used to derive empirical values are explained, along with some excursions into alternative ideas and a brief analysis of the results.

2. Spread Spectrum Image Steganography

Spread Spectrum Image Steganography (SSIS) works by storing a message as Gaussian noise in an image (Marvel, Boncelet, and Retter 1998, Marvel et al. 1999). At low noise power levels, the image degradation is undetectable by the human eye, while at higher levels the noise appears as speckles or "snow." The process consists of the following major steps, as illustrated in figure 1:

1. Create encoded message by adding redundancy via error-correcting code.
2. Add padding to make the encoded message the same size as the image.
3. Interleave the encoded message.
4. Generate a pseudorandom noise sequence, n.
5. Use encoded message, m, to modulate the the sequence, generating noise, s.
6. Combine the noise with the original image, f.

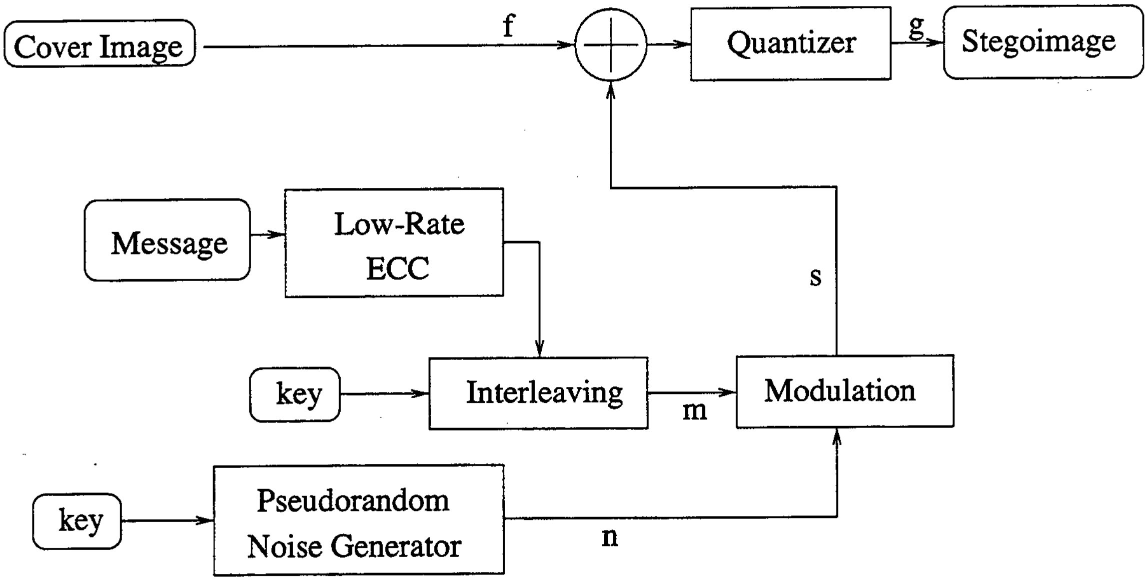


Figure 1. Simplified Steganography Embedder.

Figure 2 shows the decoding process. Notice that the original image is *not* required to recover the hidden message. A filter is used to extract the noise from the stegoimage, resulting in an approximation of the original image. The better this filter works, the fewer errors in the extracted message. This is discussed further in section 6.2.

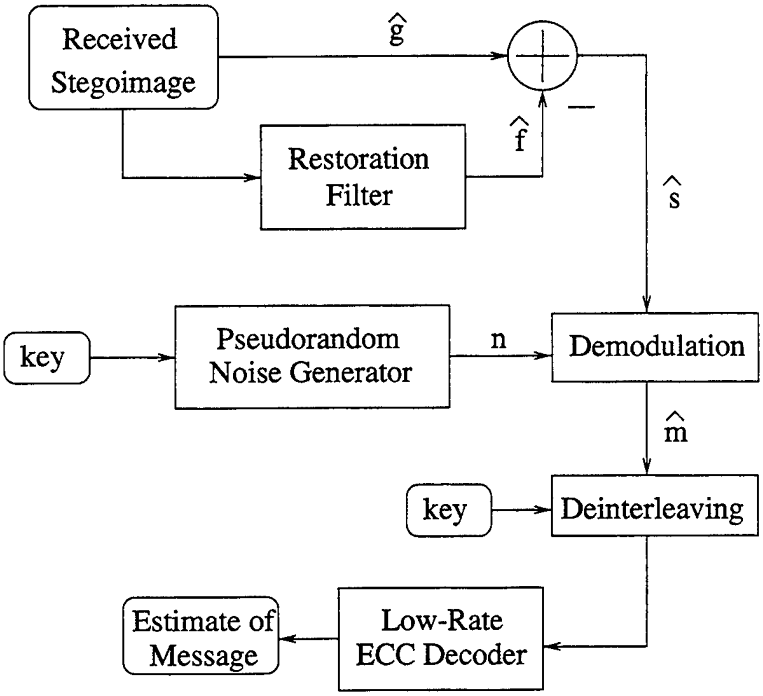


Figure 2. Simplified Steganography Extractor.

The reverse process, of extracting and restoring the original message, is of course very similar:

1. Filter the stegoimage, g, to get an approximation of the original image, $\hat{f}$.
2. Subtract the approximation of the original image from the stegoimage to get an estimate of the noise, $\hat{s}$, added by the embedder.
3. Generate the same pseudorandom noise sequence, n.
4. Demodulate by comparing the extracted noise with the regenerated noise.
5. Deinterleave the estimate of the encoded message, $\hat{m}$, and remove the padding.
6. Use error-correcting decoder to repair the message as needed.

3. System Components

3.1 Overview

During the initial research phase of the SSIS work, each major step was performed by a different component. The image manipulation had been done with MATLAB, an integrated technical computing environment that combines numeric computation, advanced graphics and visualization, and a high-level programming language (MathWorks 1999). When this prototype was built, the operations that had been performed with MATLAB were rewritten in C for portability. The discussion that follows gives a summary of each component and its operation.

3.2 Error-Correcting Code (ECC)

A family of ECC programs was written (in the C programming language) to read a message file and insert error-correcting codes. Each program uses a different set of codewords to enable it to correct a specific bit error rate (BER) (Retter 1995). The BER is computed as a percentage that represents the number of bits in error divided by the total number of bits. For the block codes used here, the BER is computed on a block-by-block basis with a block equivalant to a segment of bits that is equal to the number of input bits.

There is a tradeoff between the BER correcting capability and the message payload. This can be related to the the rate of ECC, which indicates the ratio of the number of input bits to the output bits. Typically, codes that have a low rate (less than 0.3) can correct a significant number of errors but at the cost of a large overhead. Use of a low rate code results in a low payload throughput. A higher rate ECC, on the other hand, may not correct as many errors but incurs less overhead and thereby has a higher payload throughput. Four different sets of codewords were generated to cover a range of BERs.

3.3 Interleaver

Gaussian noise is added to the image by having each bit in the encoded message alter each pixel in the image. This requires that the message and image be the same size, so the encoded message is padded with zeroes. Experimentation has shown that filtering to remove the noise from the stegoimage at the receiver causes errors typically to occur in bursts, usually correlating to the edges in the image.

The ECC works much better if the errors are not clustered together since the ECC BER is computed on a block basis. Therefore, the ECC has a better opportunity to decode correctly when the errors are spread uniformly throughout the entire image. To minimize the chance that an encoding block will have more errors than it can correct, the encoded message bits are *interleaved* or redistributed throughout the message. This act causes the number of errors in any encoding block to occur in an equally likely fashion.

3.4 Message Embedder

Another C program, based on code developed in MATLAB, was written to combine the cover image with the modified message. The user provides a key and the variance (power) of the noise. In order to store a large payload, a high rate ECC must be used, which in turn requires a low BER. This may be achieved by employing a larger power, but that degrades the quality of the stegoimage. A series of debug flags may be set when the program is run to record the computations and verify that everything is working properly. The embedder was validated by examining the results at each stage.

A stegoimage must be independent of the computer architecture that was used to create it. A standard graphics format was used for the images, and the byte-oriented algorithm used by the ECC avoided architectural byte-ordering problems.* The only architectural issue involved the code to generate the pseudorandom noise. The library functions available on the UNIX and Windows platforms were drastically different and did not even return the same size numbers, much less the same sequence. The pertinent functions were extracted from the GNU† C library (Free Software Foundation 1996). Tests showed that a Sun and a PC generated the same sequence when given the same key.

3.5 Message Extractor

During the early stages of the work, the message extractor was a separate program. Since the embedder and extractor share much of the same code, they were later combined into a single program. This had the unplanned benefit of allowing various experiments to be performed as shown in section 6.

The user still provides the key for the random number sequence in order to extract the message. He may also specify which one of several built-in restoration filters is used on the stegoimage. To avoid constantly adding new filters to the program and to make it easy to test filters for which source code was not readily available, an option was included to let the user provide a prefiltered image. In figure 2, the Restoration Filter box is removed, and the user supplies the filtered image, $\hat{f}$, directly.

3.6 Error Maps

The extracted message may have errors because the restoration filter may not perfectly generate the original cover image. These errors often occur along edges in the image. Even when the cover image was supplied as the filtered image during testing, the message was not always correct. The reason for this is truncation errors caused when noise was added to an image pixel resulting in a value less than 0 (black) or greater than 255 (white).

A Perl script (Wall, Christiansen, and Schwartz 1996) was written to compare an original and extracted message and count the number of bits that are incorrect; in other words, it computes the BER. It also creates a new image, where a white pixel indicates the message bit was correctly extracted and a black pixel indicates an error. These images, called error maps, showed the errors are not random and occur along edges and white or black areas in the image as shown later.

*The Sun used to develop the software is big-endian, while Intel chips are little-endian.

†The GNU project includes a portable C compiler and library functions.

3.7 Deinterleaver and ECC Decoder

The bits that make up the extracted message are restored to their original positions by reversing the interleaving process. The process is completed by running the extracted and deinterleaved message through the C program that matches the original code (i.e., uses the same codewords). The final result is a file that contains the recovered message.

4. Prototype System

4.1 Overview

A program was written in Java 1.2 (Flanagan 1999) to allow anyone to run all of the above components without being aware of the details. It allows the user to choose a cover image from a group of thumbnail images, select a text file, and create a stegoimage. They may also browse thumbnails of stegoimages (images with messages hidden in them), pick one, and extract the original message. Because the prototype system is GUI-based, much of the explanation that follows is in the form of screen dumps.

4.2 Hiding a Message

When the prototype is first started, it reads and stores all of the sample text files along with the message sizes and titles. It then presents the user with a simple prompt to "Select an option from the Command menu." The menu is shown in figure 3 with the user about to select **Create Stego Image.**

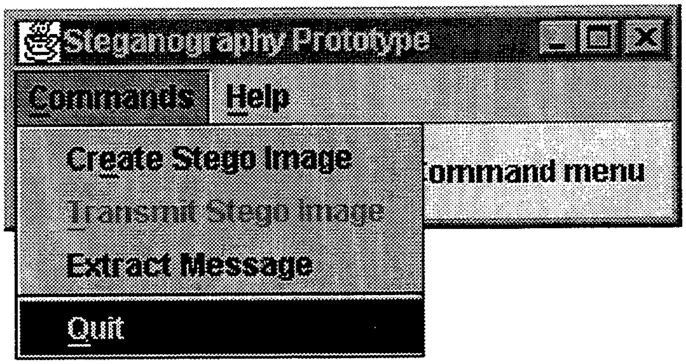


Figure 3. Command Menu.

Once the user has entered the stegoimage creation mode, the main window displays two empty panes. The one on the left is for the cover image, while the right-hand pane is a scrollable area for the display of the message to be hidden. A button under each pane allows the user to select the files he wants. Figure 4 shows the image browser with the sample set of cover (or "plain") images, while figure 5 shows the text browser on top of the main window. The number after each message title is the size of the message in characters.

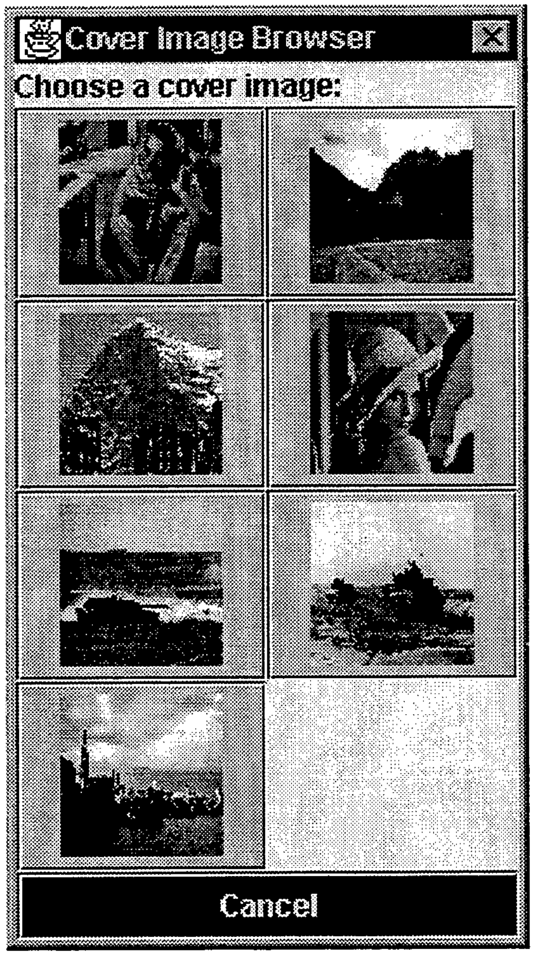


Figure 4. Image Browser.

In figure 6, the user has chosen both a cover image and a text message. The **DoIt** button appeared, the user clicked it, and now a dialog is asking the user to enter a password to be used as the embedding key. The password is converted into an integer as required for the random number generator, or the user may enter an integer key directly.

The image, message, key, and some other values are passed to the C programs that insert the ECC and create the stegoimage. The original image and the stegoimage are displayed so the user may see the degradation in image quality. This is shown in figure 7. The message file is saved for later comparison with the extracted message so that the errors may be displayed and the BER computed.

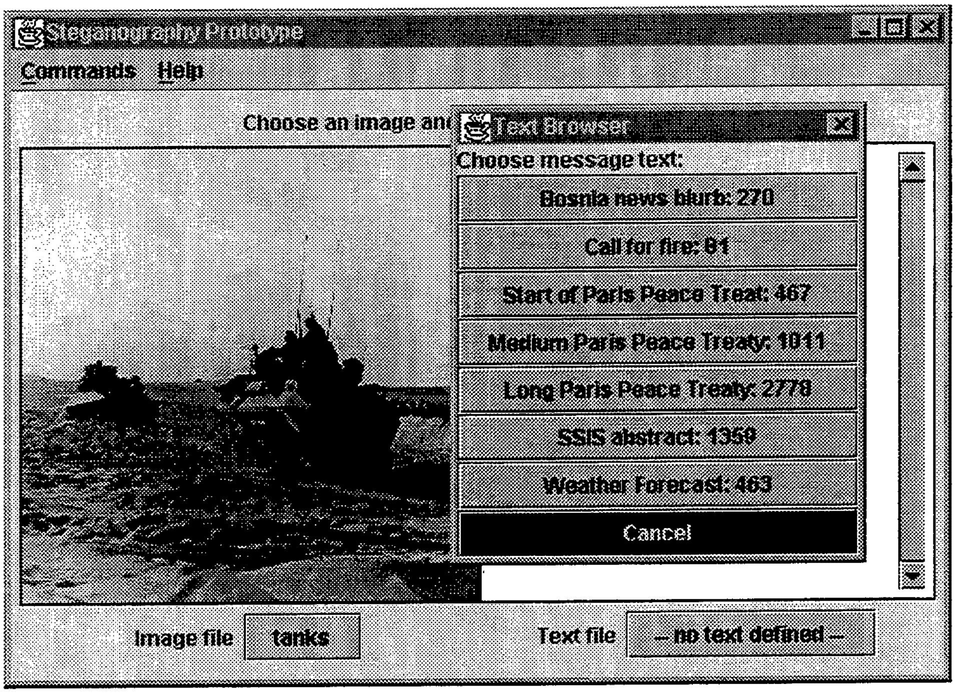
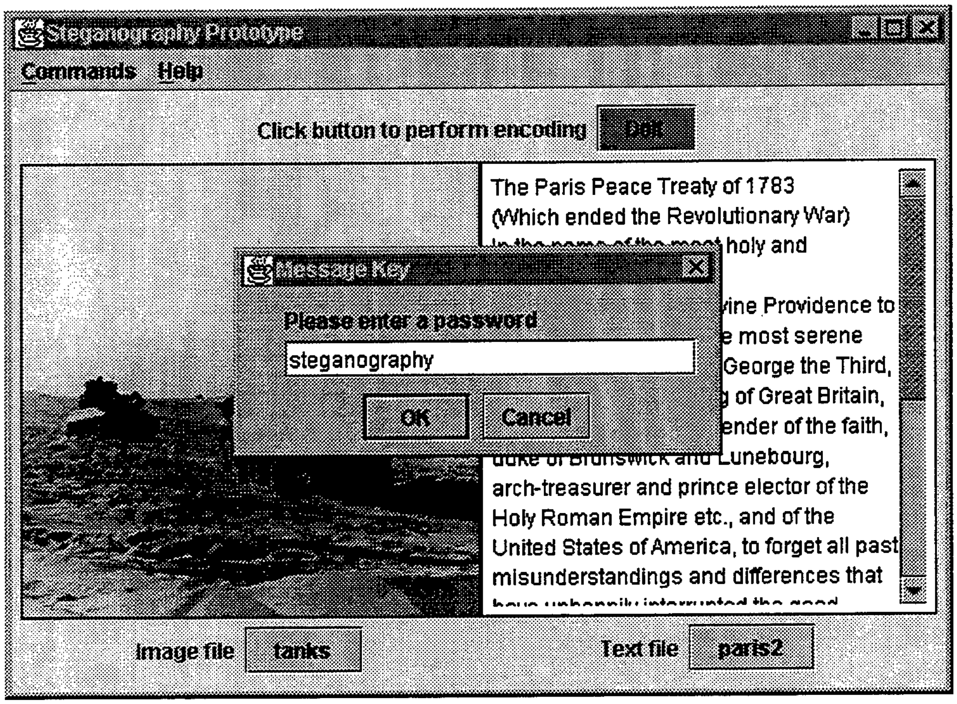


Figure 5. Using Text Browser After Selecting Cover Image.

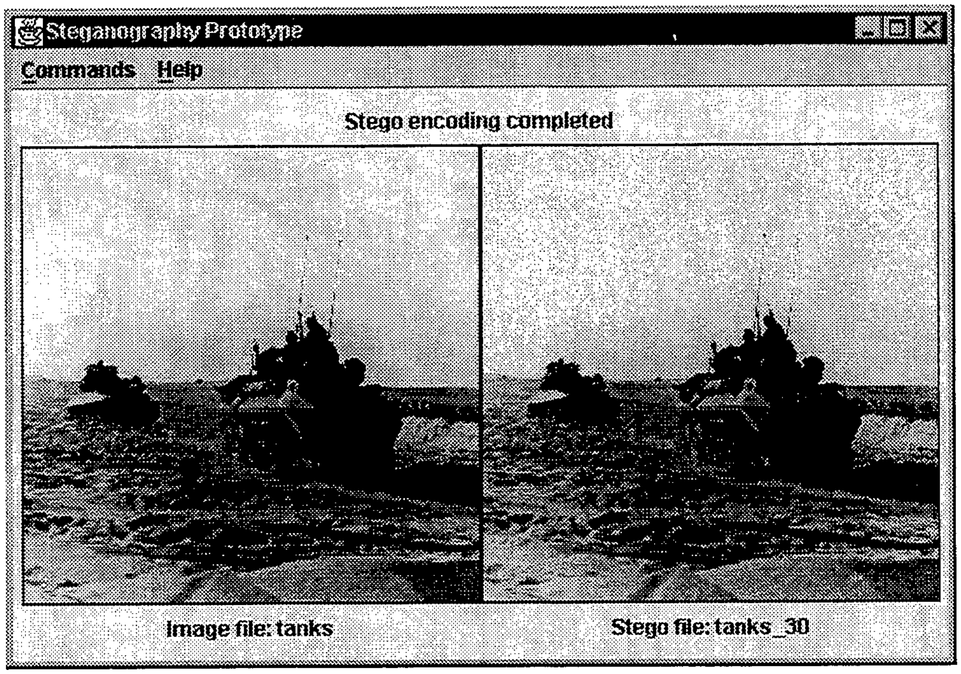


The Paris Peace Treaty of 1783
(Which ended the Revolutionary War)
In the name of the most holy and
ine Providence to
e most serene
George the Third,
g of Great Britain,
ender of the faith,
duke of Brunswick and Lunebourg,
arch-treasurer and prince elector of the
Holy Roman Empire etc., and of the
United States of America, to forget all past
misunderstandings and differences that

Please enter a password

steganography

Figure 6. Entering Message Key Prior to Embedding.



Stego encoding completed

Image file: tanks

Stego file: tanks_30

Figure 7. Original Image and Stegoimage.

4.3 Extracting a Message

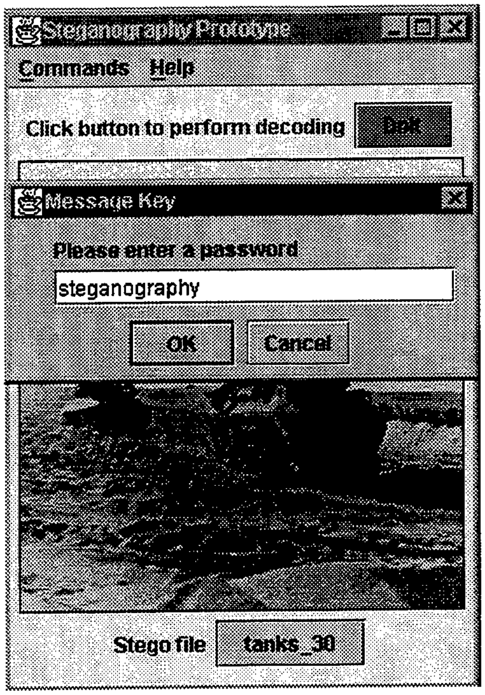
The sequence of steps to recover a hidden message from a stegoimage begins with the user selecting **Extract Message** from the command menu, which causes the main window to display a single, empty pane. Clicking on the **Stego file** button opens an image browser like the one in figure 4, only this time it displays thumbnail images of the stegoimage files. Once again, clicking the **DoIt** button causes the password dialog to appear as shown in figure 8.

The C programs manipulate the stegoimage—extracting, deinterleaving, and correcting the message. The error map* and final message are both displayed, and the process is complete. This is demonstrated in figure 9. Notice that the sky contains very few errors, while the sharp lines caused by the vehicle antennae are clearly visible, caused by the inability of the current filter to remove noise in the edge areas. The text in the right-hand pane has some errors, indicating the error-correction scheme did not repair all of the errors.

5. Implementation Details

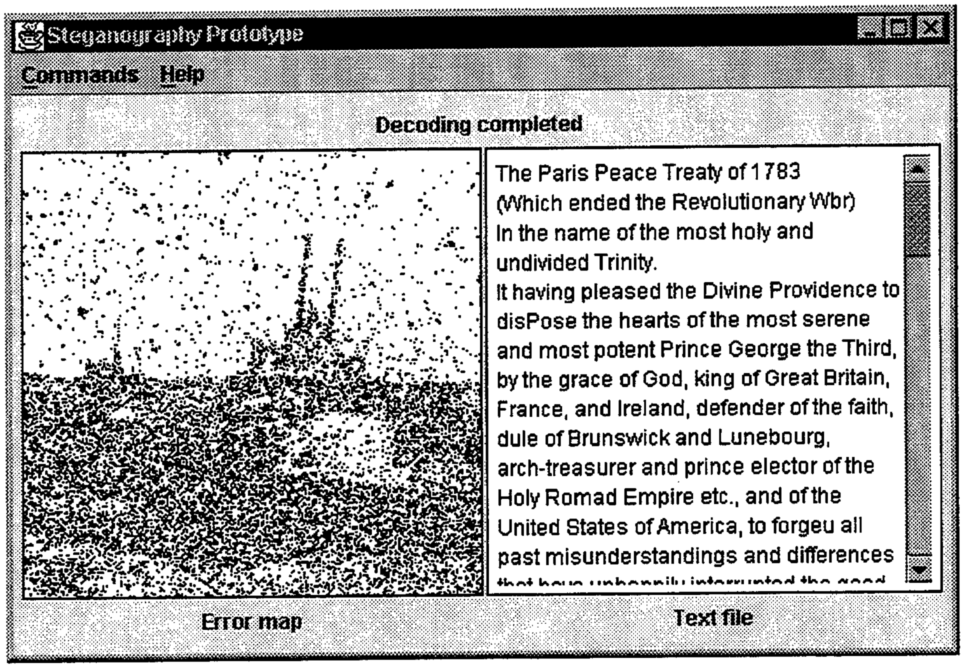
The prototype steganography system must run with a minimum of user intervention. The ECC programs currently support four sets of codewords as shown in table 1. More correcting code must be added as the error rate increases (becomes worse), which reduces the size of

*The error map is not required, but is displayed to show that the estimated message contains errors that must be corrected.



steganography

Figure 8. Entering Message Key Prior to Extraction.



The Paris Peace Treaty of 1783
(Which ended the Revolutionary Wbr)
In the name of the most holy and
undivided Trinity.
It having pleased the Divine Providence to
disPose the hearts of the most serene
and most potent Prince George the Third,
by the grace of God, king of Great Britain,
France, and Ireland, defender of the faith,
dule of Brunswick and Lunebourg,
arch-treasurer and prince elector of the
Holy Romad Empire etc., and of the
United States of America, to forgeu all
past misunderstandings and differences
that have unhappily interrupted the good

Figure 9. Error Map and Extracted Message.

the payload. The payload in bits per pixel is the payload divided by the ECC rate or bit ratio (e.g., 40/155 = 0.2581). The last column—payload capacity in characters—is based on the 256 × 256 pixel cover images used in the prototype.* If the text message is compressed, the overall capacity may be increased, but compression schemes may be fragile and may fail when the data contains errors.

Table 1. Error-Correcting Codes

Binary Code	BER Correcting Capability	Payload (bpp)	Payload (char)
(155,40)	0.12	0.2581	2114
(378,36)	0.21	0.0952	779
(889,35)	0.27	0.0393	321
(2040,32)	0.34	0.0156	127

The BER for various stego powers must be computed in order to determine the power required for each payload size. A message was hidden into the same cover image using a variety of powers. Each was then extracted and the BER computed by comparing the extracted (but not error-corrected!) message with the original message. Table 2 contains the error rates for two of the sample images.

The goal is to use the smallest power to hide a message, thus minimizing the amount of noise in the stegoimage, which corresponds to its visual appearance. These values are emboldened in table 2. Notice that Image 1 achieves all four error rates and thus may embed a message up to each specified size. Image 2 starts at a lower BER but does not get much better, so it is incapable of hiding a message of 2114 characters. The BER is always higher than the desired value of 12%. A power of 150 has an error rate of 16%† and the picture quality is becoming unacceptable.

A payload capacity file was created for each of the cover images. It contains pairs of numbers listing the threshold powers and message sizes that were determined empirically. When the user selects a cover image and a message, the image's capacity file is read to determine what power should be used to generate the stegoimage. If the message exceeds the capacity of the image, an error message is displayed and the user is prompted to choose a different message or cover image.

When the user wishes to extract the message from a stegoimage, he must supply the same key which was used as the embedding key. Normally, he would also give the power of the noise. Since this value was provided during the embedding process without his knowledge, it is incorporated into the name of the stegoimage file. In a fielded system, the user would be told what power had been used to generate the image, and this value would somehow be conveyed, as part of the key, to the person who performs the extraction.

*For example, (0.2581 × 256 × 256)/8 = 2114.36 hidden characters.

†Increasing the stego power results in smaller improvements of the BER, approaching a limit of just under 16%.

Table 2. **BER as a Function of Power**

	Image 1		Image 2	
Power	BER	Payload	BER	Payload
5	**28.5690**	**127**	**25.3815**	**321**
10	**23.7106**	**321**	23.2391	321
15	21.2021	321	22.0413	321
20	**19.3634**	**779**	21.1685	321
25	18.1107	779	**20.5292**	**779**
30	17.1417	779	20.0500	779
35	16.2643	779	19.6060	779
40	15.5838	779	19.2245	779
45	14.8941	779	18.8339	779
50	14.4165	779	18.5989	779
55	13.8596	779	18.3060	779
60	13.4476	779	18.0389	779
65	13.1241	779	17.8650	779
70	12.8143	779	17.6682	779
75	12.4878	779	17.5140	779
80	12.2711	779	17.3676	779
85	**11.9614**	**2114**	17.2623	779
90	11.7783	2114	17.1204	779
95	11.5616	2114	17.0166	779
100	11.3571	2114	16.9357	779

6. Laboratory Excursions

The modular design of the prototype steganography system makes it very easy to conduct experiments. Because the embedding and extraction of a message in a stegoimage is performed in a single program, it is very easy to share code.

The Java front end was not used in the laboratory tests because it limits the user's choice of parameters. The various programs were either invoked manually or with an automated script. This is work in progress, so general observations are made, but a detailed analysis of the results is not possible at this time.

6.1 Image Estimation

The first step in extracting a message from a stegoimage consists of applying a restoration filter to the stegoimage. If the filter perfectly reconstructs the original cover image, the embedded message should be recovered with no errors. However, the extracted message sometimes contained errors caused by truncation, as explained in section 3.6.

The better the filter works, the fewer errors there are in the extracted message. A stegoimage was created and subjected to a variety of filtering programs. The filtered stegoimages were then supplied to the message extraction program, and the BER was computed for each one. In this way a number of filters were (and continue to be) evaluated, and a few were incorporated into the steganography program.

6.2 Prefiltering

Edges that appear in an image are a major source of extraction errors. The edges in figure 10 are readily apparent; both the outline of the mountain and the window frames on the building are clearly visible. An experiment was conducted where a filter was applied to the cover image *before* the message was added to it. This had the effect of smoothing out the edges, and the BER of the extracted message was reduced dramatically as shown in table 3. However, the stegoimages produced had a blurry, slightly out-of-focus appearance and were deemed visually unacceptable at this time.

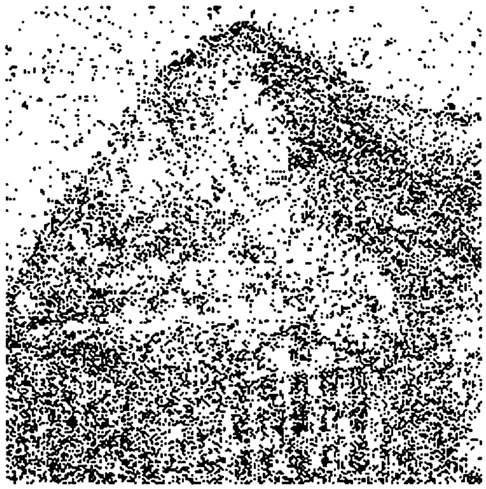


Figure 10. Error Map With Obvious Edges.

Table 3. BER for Prefiltered Images

Cover	BER (%)	
Image	Normal	Prefiltered
A	19.28	6.78
B	24.24	10.29
C	12.81	6.43

6.3 Stegoimage Modification

A related idea approaches the filtering problem from a different direction. Instead of finding a better filter or blindly prefiltering the cover image, take the extraction process into account. Generate the stegoimage the usual way, then immediately extract the message. The original message is still available, and errors may be readily detected.

The embedder may be modified to call the appropriate extraction functions and use the errors as immediate feedback to make corrections. Various adaptive techniques are being explored, and early results have shown that the BER may be reduced significantly with little effect on the quality of the stegoimage.

7. Future Research

SSIS is made up of several components, and each of them may be enhanced to improve the entire process. For example, one of the ECC codeword sets has been replaced with a set that fixes more errors, and a fifth ECC scheme is being tested. Another idea is to add some intelligence to the error-correcting process. It is known that edges cause problems, so some form of edge detection must be used to augment the ECC algorithm (Marvel and Retter 2000).

A solution to the truncation problem may lie in scaling the data so that all pixel values are in the range of the original image. Preliminary tests show that adding a feedback loop to the embedding process reduces extraction errors without degrading the stegoimage, although care must be taken to ensure the image is not altered in an unacceptable manner. A filter that produces mediocre results with the standard algorithm may turn out to be much better when feedback is included. All changes made to the stegoimage must be transparent to the extractor; no out-of-band data may be exchanged between the sender and recipient (besides the key and stego power) before the message is extracted.

A more elaborate approach is to automate the entire process. The current prototype requires the user to select the cover image for his message, and the power used in the embedding is based on a fixed table of message sizes. The embedding and extraction portions of the SSIS process are fairly quick, and it may be possible to iterate on a range of powers to determine the best to use for a given message and image combination. An algorithm to determine picture quality, by comparing the cover image and stegoimages, may be added to automatically determine if a power results in an acceptable image.

8. Summary

SSIS has been shown to be a powerful way to transmit messages via normal channels without an observer detecting them. The prototype system described in this report is both functional and portable, using languages available on a variety of platforms. The modular design of the system allows components to be replaced with more powerful ones and lends itself to casual experimentation by multiple researchers. Some of the "what-if" excursions that have been conducted were not anticipated during the development phase of SSIS. More experiments are expected as the process is refined further.

INTENTIONALLY LEFT BLANK.

9. References

Flanagan, D. *Java in a Nutshell.* Sebastopol, CA: O'Reilly & Associates, 1999.

Free Software Foundation. GNU C Library. http://www.gnu.org/software/libc/, 1996.

Marvel, L. M., C. G. Boncelet, Jr., and C. T. Retter. "Methodology of Spread-Spectrum Image Steganography," ARL-TR-1698, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, June 1998.

Marvel, L. M., C. G. Boncelet, Jr., and C. T. Retter. "Spread Spectrum Image Steganography," *IEEE Transactions on Image Processing*, Vol 8, No 8, pp 1075–1083, August 1999.

Marvel, L. M. and C. T. Retter. "The Use of Side Information in Image Steganography," To be presented at the 2000 International Symposium on Information Theory and Its Applications (ISITA'2000), Honolulu, HI, 5–8 November 2000.

MathWorks. MATLAB. http://www.mathworks.com/products/matlab, 1999.

Retter, C. T. "Binary Weight Distributions of Low Rate Reed-Solomon Codes," ARL-TR-915, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, December 1995.

Wall, L., T .Christiansen, and R. L. Schwartz. *Programming Perl.* Sebastopol, CA: O'Reilly & Associates, 1996.

INTENTIONALLY LEFT BLANK.

NO. OF COPIES	ORGANIZATION
2	DEFENSE TECHNICAL INFORMATION CENTER DTIC DDA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1	HQDA DAMO FDT 400 ARMY PENTAGON WASHINGTON DC 20310-0460
1	OSD OUSD(A&T)/ODDDR&E(R) R J TREW THE PENTAGON WASHINGTON DC 20301-7100
1	DPTY CG FOR RDA US ARMY MATERIEL CMD AMCRDA 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001
1	INST FOR ADVNCD TCHNLGY THE UNIV OF TEXAS AT AUSTIN PO BOX 202797 AUSTIN TX 78720-2797
1	DARPA B KASPAR 3701 N FAIRFAX DR ARLINGTON VA 22203-1714
1	US MILITARY ACADEMY MATH SCI CTR OF EXCELLENCE MADN MATH MAJ HUBER THAYER HALL WEST POINT NY 10996-1786
1	DIRECTOR US ARMY RESEARCH LAB AMSRL D D R SMITH 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB AMSRL DD 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB AMSRL CI AI R (RECORDS MGMT) 2800 POWDER MILL RD ADELPHI MD 20783-1145
3	DIRECTOR US ARMY RESEARCH LAB AMSRL CI LL 2800 POWDER MILL RD ADELPHI MD 20783-1145
1	DIRECTOR US ARMY RESEARCH LAB AMSRL CI AP 2800 POWDER MILL RD ADELPHI MD 20783-1197
	ABERDEEN PROVING GROUND
4	DIR USARL AMSRL CI LP (BLDG 305)

NO. OF COPIES	ORGANIZATION
	ABERDEEN PROVING GROUND
19	DIR USARL AMSRL CI DR N RADHAKRISHNAN DR J GANTT AMSRL CI C DR J GOWENS AMSRL CI CN H HARRELSON (4 CPS) AMSRL CI CT F BRUNDICK (5 CPS) G HARTWIG DR L MARVEL (5 CPS) DR C RETTER

NO. OF COPIES	ORGANIZATION

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project(0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 2001	3. REPORT TYPE AND DATES COVERED Final, June 1999–August 2000

4. TITLE AND SUBTITLE
Implementation of Spread Spectrum Image Steganography

5. FUNDING NUMBERS
AH480TEP20

6. AUTHOR(S)
Frederick S. Brundick and Lisa M. Marvel

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
ATTN: AMSRL-CI-CT
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-2433

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT*(Maximum 200 words)*

Steganographic techniques are useful to convey hidden information by using various types of typically-transmitted multimedia data as cover for concealed communication. Spread Spectrum Image Steganography (SSIS) is a data-hiding/hidden-communication method that uses digital imagery as a cover signal. This report describes an SSIS prototype system for embedding messages in images and extracting messages from stegoimages. The components, which were written in the Java and C programming languages, were kept modular to provide a workbench for further experimentation. We discuss experiments that were performed with the system, along with possible avenues of future research to improve the SSIS process.

14. SUBJECT TERMS
steganography, capacity, hidden communication, implementation

15. NUMBER OF PAGES
23

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

INTENTIONALLY LEFT BLANK.

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number/Author ARL-TR-2433 (Brundick) Date of Report March 2001

2. Date Report Received

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.)

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.)

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate.

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.)

CURRENT ADDRESS

Organization

Name E-mail Name

Street or P.O. Box No.

City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

Organization

Name

Street or P.O. Box No.

City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)

(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 0001, APG, MD

POSTAGE WILL BE PAID BY ADDRESSEE

DIRECTOR
U.S. ARMY RESEARCH LABORATORY
ATTN AMSRL CI CT
ABERDEEN PROVING GROUND MD 21005-5067